Master's Thesis

Prediction-based Dynamic Capacity Allocation for Traffic Cost Minimization

Ddaddt-hanbit Bae

Department of Computer Science and Engineering

Graduate School of UNIST

2020

Prediction-based Dynamic Capacity Allocation for Traffic Cost Minimization

Ddaddt-hanbit Bae

Department of Computer Science and Engineering

Graduate School of UNIST

Prediction-based Dynamic Capacity Allocation for Traffic Cost Minimization

A thesis/dissertation
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Ddaddt-hanbit Bae

12/30/2019

Approved by

Advisor

Prediction-based Dynamic Capacity Allocation for Traffic Cost Minimization

Ddaddt-hanbit Bae

This certifies that the thesis/dissertation of Ddaddt-hanbit Bae is approved.

12.30.2019

signature

Advisor: Kyunghan, Lee

signature

Changhee Joo

signature

Hyunjong Yang

Abstract

Recent advances in network virtualization techniques have shed light on dynamic resource allocation according to traffic usage. In particular, the minimum total network usage cost is achievable by using on-the-fly capacity allocation with accurate traffic estimation. In practice, there is an unavoidable delay for system reconfiguration, and thus a precise prediction on the traffic usage is required, which is, however, challenging due to unexpected system dynamics such as mobility and time-varying wireless channels. In this work, we address the prediction-based capacity allocation to minimize traffic cost by exploiting deep learning techniques. We develop an MLP model for accurate prediction of traffic usage, which is trained with real-world system logs obtained in a firewall. Taking into account the prediction errors and asymmetric structure of capacity pricing, we develop an efficient online capacity allocation scheme that achieves low traffic cost. We also evaluate the performance of our solution using the real-world data.

Contents

Figure Contents

I. INTRODUCTION

The Internet has been evolved with a number of modern communication technologies in modulation, media access, queuing [1], security, traffic service [2]. Recently, as artificial intelligence (AI) technologies rapidly advances, it is questioned whether one can apply machine learning techniques to the aforementioned conventional network problems. This paper aims to take this approach in lowering network cost, in particular, the operation cost of enterprise network through dynamic capacity allocation. Conventionally, a static network capacity is allocated through a contract between the enterprise and an Internet service provider (ISP). Nowadays, with an advance in switching technology, it can be dynamically adjusted based on real-time capacity request.

In the conventional method, where an enterprise makes a contract for the upper limit of fixed capacity with periodic payment to the ISP, the contracted bandwidth is the maximum level that the enterprise is expected to use over the agreed period. The maximum capacity is usually set based on their experience. Mostly, the capacity is not fully used at the peak rate, and the network is under-utilized, which can be considered as the overhead. In contrast, when the traffic exceeds the estimated maximum level, the performance will be degraded due to packet drops, queuing, etc.

As the switching technology advances, ISPs start providing flexible capacity services [3] [4]. It is possible for ISPs to dynamically manipulate system configurations, and changes the allocated network capacity upon request [11]. New recent techniques such as cloud and SDN can be also used to support this functionality [12].

In this new environment, the enterprise can exploit flexible capacity to reduce the connectivity cost: request a larger capacity when the traffic demand is high, and reduce the capacity when the traffic demand is low. One may design a scheme that estimates time-series network traffic, and optimizes the network expense by requesting appropriate capacity from the ISP in a timely manner. However, since it takes an unavoidable and non-negligible time to change the network system configuration, it is imperative to make an accurate prediction of time-series network traffic to avoid either performance degradation or unnecessary cost. In this work, we consider the flexible capacity service with asymmetric price structure: when the traffic demand is smaller than the requested capacity, the ISP charges a fixed per-capacity price, and when the traffic demander is greater than the requested capacity, the excessive traffic is accommodated but at higher per-traffic price. We aim to minimize total cost by predicting the traffic demand with accuracy, and by taking into account prediction error and the asymmetric price structure. The main contributions of this work are as follows.

- We develop a scheme that predicts Internet traffic using only Firewall log, which removes additional equipment for traffic prediction and substantially facilitates the deployment of our solution.
- We design a deep-learning based scheme that predicts the time-series traffic with accuracy.
- We minimize the total cost of capacity allocation accounting for the asymmetric price structure under prediction error.

The paper is structured as follows. We provide a brief review on the previous works in Section II. After we specify our system model and formulate the problem in Section III, we tune our prediction model and proposed schemes that account for the asymmetric pricing structure in Section IV. Evaluating our solution in Section V, we conclude our work in Section VI.

II. RELATED WORKS

Deep learning for time-series forecasting has been studied at the frontlines to understand many social, economic, natural, and IT phenomena. Network traffic prediction, which this paper focuses on, has been also studied in many different ways [14]. In this paper, traffic prediction refers to forecasting the total amount of network traffic in the next short period of time. This task can generally be classified into two different groups: to use statistical methods and to use artificial neural network.

First, several statistical techniques including Linear Regression (LR) [7], Logistic Regression and Moving Average like Auto Regressive Moving Average (ARMA) [8] have been used to predict network traffic. Linear regression analysis proceeds according to the target regression function that is either linear, quadratic, or high-polynomial. The logistic regression method is similar to LR but it uses a logistic function. These methods share the difficulty in identifying dependent / independent variables when the prediction becomes complex. Prediction by regression analysis only adjusts the variables that map a particular sequence, without reflecting the predictions for the target system. In regression analysis, a small change in the underlying environment often requires new learning and tuning of the parameters, which make them hard to use in practice. ARMA is one of the traditional methods to make statistical predictions, and it derives other well-known methods such as Auto Regressive Integrated Moving Average (ARIMA) [16] and Winter-holt method. However, in some case, they need frequent re-estimation to predict and are not practical in online learning [16].

There have been studies that apply neural networks to traffic prediction [5] [6] [9] [15]. One of the most basic approaches is to use a multilayer perceptron (MLP) [6] [15]. It has been shown that a 3-layer neural networks can be implemented in hardware equipment to predict the amount of traffic for video transmission in ATM networks [9]. Another study has employed Autoencoder, and achieves 8% better prediction performance than MLP [6].

Google researchers have also used prediction method in data centers [10]. They focused on Power Usage Efficiency (PUE), and developed a 3-layer MLP model that learns 16 input variables. They designed the model such that the input variables complemented each other to mitigate prediction failures at outlier values, e.g., peak traffic.

III. PROBLEM FORMULATION

This section formulates the problem and describes how the traffic prediction can minimize the cost associated with network capacity allocation. Consider an enterprise network that subscribes flexible capacity service for Internet connectivity from an ISP. Time is slotted. Overall procedure is shown in Fig. 1 and can be described as follows. At the beginning of each time slot, the operator of the enterprise network predicts the traffic amount based on its observation in the past. Taking into account possible prediction error and pricing structure, it requests a certain amount of capacity. During the time slot, actual traffic occurs, and the ISP accommodate the traffic. At the end of the time slot, the actual traffic and its cost according to the pricing structure will be feedback to the operator. The procedure repeats in the next time slot.

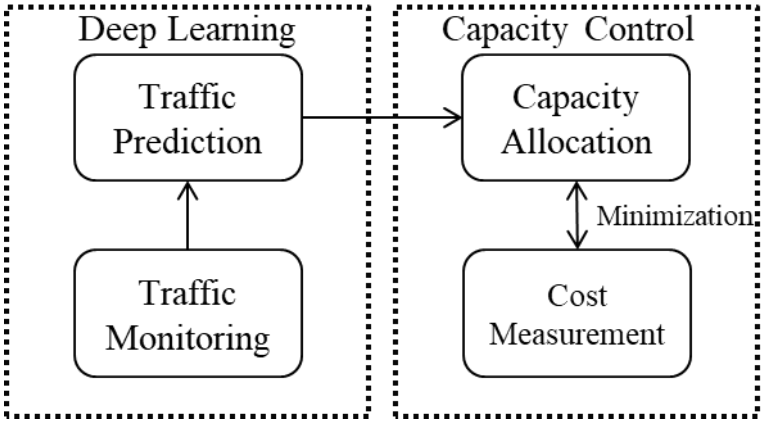


Fig. 1. Overall procedure.

In this paper, we use the Multilayer Perceptron model (MLP) for the traffic prediction. It is one of basic deep-learning models that are the fastest and the most common. An MLP internally has a set of weights with a hidden layer [13] as shown in Fig. 2.

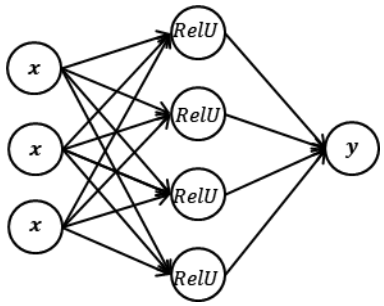


Fig. 2. Multilayer Perceptron (MLP) component.

One may employ more advanced models such as RNN or Autoencoder as in [6], which, however, did not result in substantial performance gain over MLP in our experiments. The prediction performance is often evaluated through a loss function such as mean absolute error (MAE) or mean square error (MSE). In this work, we focus on the cost associated with capacity allocation and the price structure.

We assume that the ISP charges a unit per-capacity cost. Suppose that the operator of the enterprise network predicts the traffic amount as Y_t and requests capacity C_t as predicted, i.e., $C_t = Y_t$. If actual traffic X_t in this time slot is no greater than the predicted value, then the operator will pay C_t that includes overhead $(Y_t - X_t)$. In contrast, if the prediction fails and $X_t > C_t$, then the ISP accommodates the excessive traffic $(X_t - C_t)$ at high per-traffic cost p (> 1), resulting in the cost in this time slot $C_t + p(X_t - C_t)$. We denote this penalty price as the *asymmetric pricing structure*. Letting $cost(X_t, Y_t)$ denote the cost in time slot t, we define

$$cost(X_t, Y_t) = \begin{cases} C_t, & if\ X_t \le C_t, \\ C_t + p(X_t - C_t), & if\ X_t > C_t. \end{cases}$$

Note that if the operator requests different capacity, then the cost changes. One of our goals is to find appropriate allocation policy $C_t = f(Y_t)$ to minimize the total cost, i.e.,

$$\underset{C_t = f(Y_t)}{\text{minimize}} \sum_{t} \text{cost}(X_t, C_t).$$

IV. PROPOSED METHOD

We employ a deep-learning model to make accurate predictions on the next-time traffic. We use a dataset collected from a firewall log in an enterprise network, and train the model using the data.

In this section, we describe the data, and specify how the preprocessing and the model training are applied. Finally, we proposed several heuristic capacity allocation schemes.

4.1 Preparing Dataset

We collect traffic data from a firewall log in an enterprise network. Note that previous works often use traffic data obtained from a backbone switch [6], which, however, is often unavailable at the enterprise side and thus hard to be used in practice to predict the traffic from the perspective of the enterprise-network operator. In contrast, we rely on the data obtained from a firewall in the enterprise network, which is not only accessible from the operator but also does not incur additional cost for specialized equipment. The advantages of using the firewall log can be summarized as follows.

Feature 1: As most enterprises have firewalls, there is no need to introduce additional network measurement equipment.

Feature 2: It can measure both inbound and outbound traffic at the network boundary.

Feature 3: It provides accurate traffic measurement excluding any blocked traffic, which is not available at backbone switch.

Feature 4: It can provide richer information than backbone switch such as the number of sessions.

We collect the traffic log from a firewall for 10 months in an enterprise network in Korea. The following illustrate the data that we collected. The time slot for an observation is 15 min, and the data includes bytes received, bytes sent, and the number of active sessions, etc.

Day Received	Quarter Time Received	Bytes	Bytes Received	Bytes Sent	Sessions	Threats
Mon, Mar 11, 2019	2019-03-11 14:45	62575503871	41,568,991,299	21006512572	1333646	73
Mon, Mar 11, 2019	2019-03-11 15:00	72803567434	50,366,709,703	22436857731	1528871	183
Mon, Mar 11, 2019	2019-03-11 15:15	76829011687	55,655,220,486	21173791201	1776342	72
Mon, Mar 11, 2019	2019-03-11 15:30	69580187317	49,761,517,538	19818669779	1746773	42
Mon, Mar 11, 2019	2019-03-11 15:45	69666656870	46,822,020,148	22844636722	1790676	262

Table. 1. Firewall logs samples.

This study uses a Korean company's 10 months traffic log of the internet gateway firewall. The size of 1 sample is the amount of the traffic (bytes) for the next 15 minutes. Some properties of the data are shown in Table 2.

- Observation interval: 15 minutes
- Total dataset size: 28,800 observations in 10 months
- Maximum (peak) amount of traffic (received + sent) : 1,929.97 GB
- Minimum amount of traffic (received + sent) : 3.03 GB
- Average traffic (received + sent): 45.03 GB
- Max Network Users : 21,000 users

Table. 2. Dataset Properties.

4.2 Preprocessing

This dataset may include the singularity of peaks in network traffic. For example, a peak occurs when an operating system patch that is applied to all users starts. We try to exclude this irregular traffic measurement and preprocessing the data for our time-series analysis. There are several ways to manipulate the peak values. In this study, we reshape the data, focusing on removing sudden traffic increases. To elaborate, let X_t denote actual traffic amount at time slot t in our dataset, and let X'_t denote the preprocessed data. We have

$$X'_t = \begin{cases} X_{t-1}, & if\ X_t \le 3 \cdot \min\{X_{t-1}, X'_{t-1}\}, \\ 3 \cdot \min\{X_{t-1}, X'_{t-1}\}, & if\ X_t > 3 \cdot \min\{X_{t-1}, X'_{t-1}\}. \end{cases}$$

4.3 Tuning Deep-learning Model

A Deep learning model (with MLP) typically has 2 [17] or more hidden layers [18] with fully connected output layers. The model learns the weights associated with each layer according to the input-output pairs. We have the input matrix that consists of 96 sequential observations, which corresponds to 24-hour data, and the output is the traffic amount at the very next observation. This structure makes it possible to create an objective regression function that is learned through several epochs.

We divide our dataset into three sets of training, verification and test data. As shown in Fig. 3, we repeat training and verification using the training set and the verification set, and evaluate the performance using the test set. We used the following ratios: 60% of the entire Dataset for training, 20% of the entire Dataset for verification, and remainder 20% of dataset for test.

The hidden layer structure can be formulated as follows:

$$z(\boldsymbol{x}) = g(\boldsymbol{V}\boldsymbol{x})$$

$z(\boldsymbol{x})$ presents the output layer and it is converted to input of following layer. $\boldsymbol{V}$ is a weight matrix that contains effectiveness weight in the training set. And $g(u)$ is an activation function which presents activation weight when passing effective weight to consequence layer.

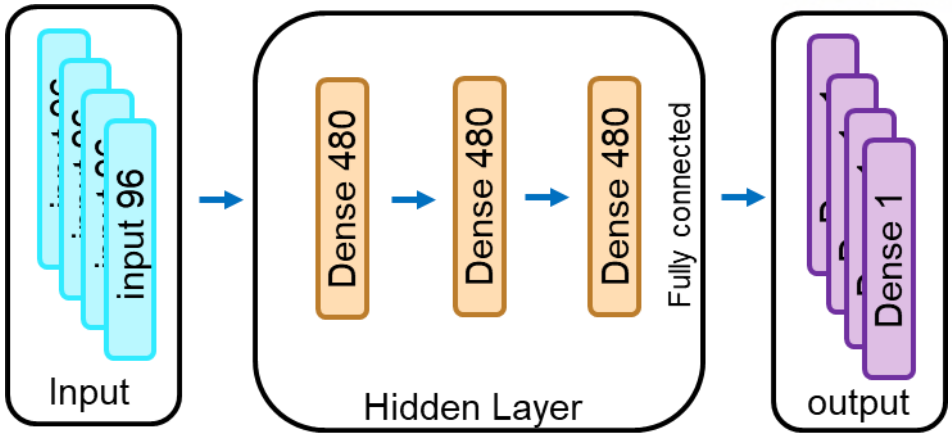


Fig. 3. Proposed Deep-learning structure.

In our model,

$$\boldsymbol{x}_2 = Z(\boldsymbol{x}_1) = g(\boldsymbol{V}\boldsymbol{x}_1), \quad matrix\ size\ \boldsymbol{x}_1: 96 \times 1, \mathbf{V}: \ 480 \times 96$$
$$\boldsymbol{x}_3 = Z(\boldsymbol{x}_2) = g(\boldsymbol{V}\boldsymbol{x}_2), \quad matrix\ size\ \boldsymbol{x}_2: 480 \times 1, \mathbf{V}: \ 480 \times 480$$
$$\boldsymbol{x}_4 = Z(\boldsymbol{x}_3) = g(\boldsymbol{V}\boldsymbol{x}_3), \quad matrix\ size\ \boldsymbol{x}_3: 480 \times 1, \mathbf{V}: \ 480 \times 480$$
$$\boldsymbol{Y} = Z(\boldsymbol{x}_4) = g(\boldsymbol{V}\boldsymbol{x}_4), \quad matrix\ size\ \boldsymbol{x}_4: 480 \times 1, \mathbf{V}: \ 1 \times 480$$
$$g(u) = ReLU(u) = \max(0, u),$$

MLP model trains the weight with Back-propagation like Stochastic Gradient Descant (SGD) and we select the loss function as MSE. The back propagation procedure is usually described as follows:

1. Initialize network randomly weights
2. Present first input vector, from training data, to the network,
3. Propagate the input vector through the network to obtain an output,
4. Calculate an error signal by comparing the actual output to the desired (target) output,
5. Propagate error signal back through the network,
6. Adjust weights to minimize overall error,
7. Repeat steps 2~7 with next input vector, until the overall error is satisfactorily small. [19]

There are several hyper parameters that we need to set.

- The number of hidden layers : Predicting with a model with too many hidden layers leads to high computational complexity and degraded the learning performance by vanishing weight. On the other hand, with insufficient number of layers, training a complex function

is not possible. In this study, we use a structure with 3 hidden layers, each of which is of size 480 [17][18].

- Activation function : There are several activation functions, we used ReLU to avoid vanishing gradient problem. ReLU featured as a rectifier activation function Because of this linearity, gradients flow well on the active paths of neurons (there is no gradient vanishing effect due to activation non-linearities of sigmoid or tanh units), and mathematical investigation is easier [20].
- Dropout rate : Traditional ANN has been limited with regulation problem. To overcome this, Dropout algorithm is adjusted. Dropout is a relatively new algorithm for training neural networks, which relies on stochastically "dropping out" neurons during training in order to avoid the co-adaptation of feature detectors [21]. In our model, a higher dropout rate is preferred for lesser structured data. It turns out that the dropout of 0.2 is the best in our data.
- Input size : Since the traffic usage has a daily pattern, we need the input data of at least 1 day for accurate prediction, which corresponds to the input size of 96. We found that the size larger than 1 day does not improve the prediction accuracy but increase computational complexity.
- Epoch : The epoch is the number of training with the entire dataset. The optimal epoch is found around 250 times.
- Batch : The batch size is the number of inputs to apply to the weight calculation. We use a batch size of 32. And we adjust to batch normalization for the model that divided with a mini-batch for training. This algorithm can help in weight vanishing problem for training.

4.4 Capacity Allocation

Due to the asymmetric pricing structure, the prediction errors, i.e., overestimation and underestimation, can have a different impact on the cost. The following results demonstrate the impact of the pricing structure when $p = 10$. Fig. 4 shows that our prediction (red line) has some errors in comparison with true traffic (blue line). We have used the aforementioned settings to train our MLP model, and requests the capacity that equals our predicted value. The results illustrate that when an overestimation occurs, i.e., when the blue line is lower than the red line, the cost (green line) becomes much higher than the opposite case.

To mitigate the cost upsurge under overestimation, it is natural to request a larger capacity than the predicted value, which provides a trade-off between the overhead from additional capacity and the cost due to prediction failure. We suggest the following set of policies, where parameter k denotes a configurable integer parameter.

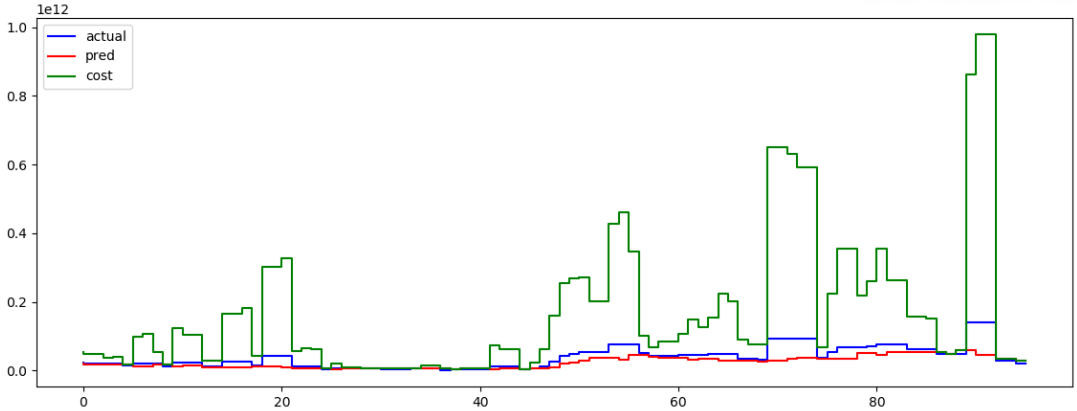


Fig. 4. Allocation capacity and its cost on 1-day.

- Naive method: $C_t = Y_t$.
- Simple multiplication (k): $C_t = Y_t\left(1 + \frac{k}{10}\right)$.
- Additional standard error (k): $C_t = Y_t + se \cdot \frac{k}{10}$, where se denotes the standard error.
- Windows standard error ($w = 3, k$): $C_t = Y_t + se_w \cdot \frac{k}{10}$, where se_w denote the standard error for the last 3 observations.
- The heuristic combination of multiplication and standard error ($w = 3, k$): $C_t = \max\left(\mathrm{Y_t} + \mathrm{se_W} \cdot \frac{\mathrm{k}}{10},\ \mathrm{Y_t} \cdot p^{1/4}\right)$

In the next section, we numerically evaluate the performance of the above schemes, and em pirically compare the schemes.

V. NUMBERICAL RESULTS

This section presents numerical results with our proposed schemes. We have trained our MLP model as described in Section IV, and allocated the capacity following 6 different methods shown in Section 4.4.

First, we allocate the capacity as the predicted value (naive method). Fig. 4 demonstrates how the predicted values (red line) are different from the ground truth (blue line). Overall, the predicted values smoothly follow the actual value. Also, we can define the prediction failure rate (PFR) as

$$prediction\ failure\ rate(PFR) = \frac{\sum |Y_t - X_t|}{\sum X_t} \times 100$$

The naive method achieves PFR about 69.2%.

If this model does not apply deep learning and capacity allocation algorithms, the traffic allocation would apply maximum traffic for the entire period as before. In this case, the average traffic cost is measured at 13.91 Gbyte. Conversely, when capacity allocation fitted perfectly

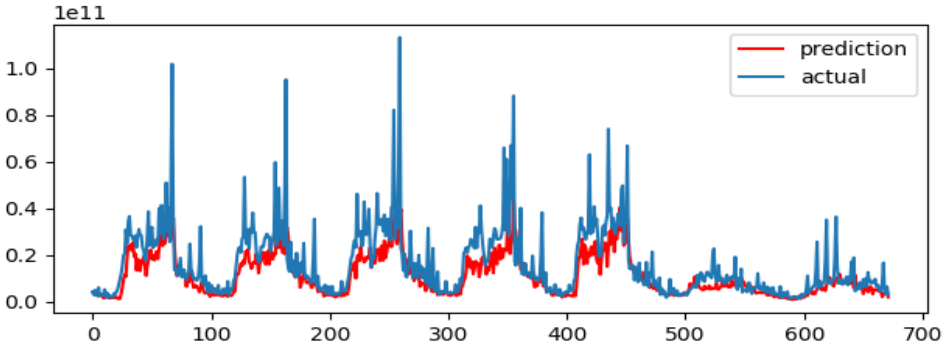
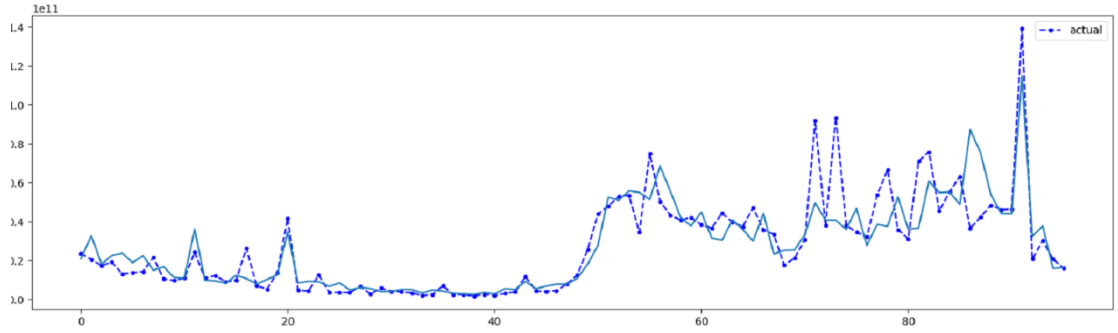


Fig. 5. Traffic prediction of our MLP model for 5 days.

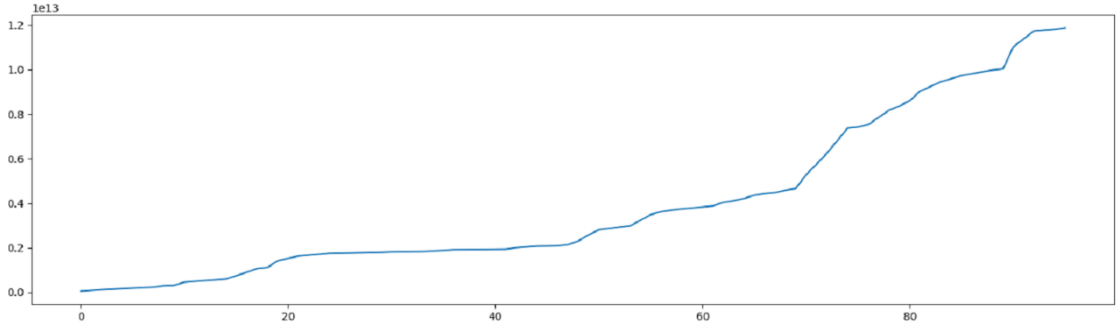
operational and prediction traffic is exactly same as actual traffic. Then there would be no overhead. The traffic is called optimal, and the average cost is about 4.81 Gbyte. In this situation, we try to minimize the network cost by applying various capacity allocations. In this problem, this paper proposes to 5 different methods.

The first, Using deep-learning without any transaction, as already mentioned, it contains 31.8% error. The average cost at this time is about 12.36 Gbyte when converted to penalty price with the asymmetric pricing structure. After analyzing this, the naive approach is not effective in reducing network costs.

Fig. 6 shows the cost after applying the capacity allocation. Due the high cost of underestimation cases, it suffers from high cost as shown in Fig. 6(b).



- Result Capacity

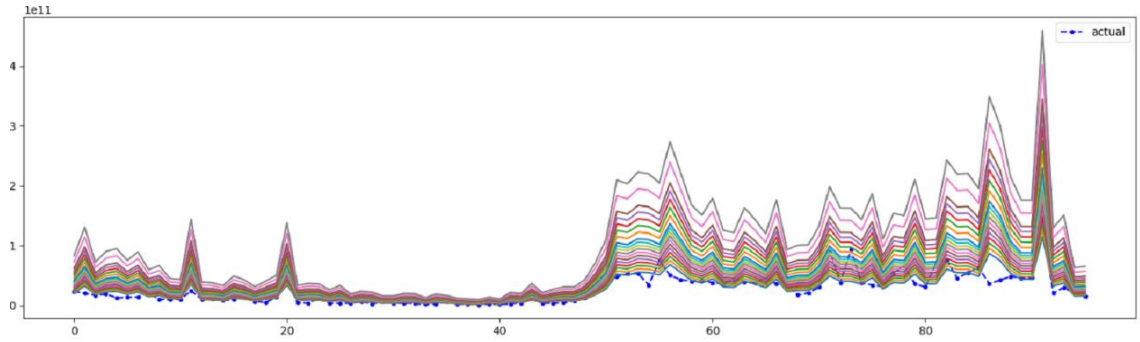


- Cost Change Flow

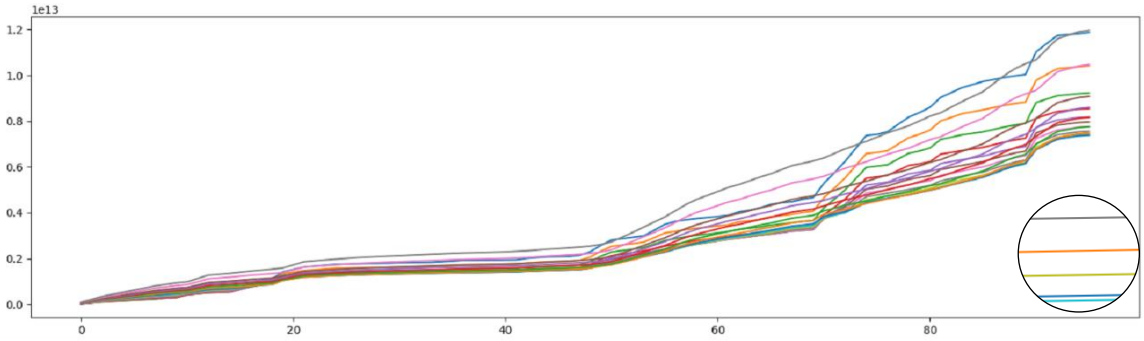
Fig. 6. Naïve method.

We devise the Simple Multiplication method. We have to consider cost increasing due to the penalty traffic cost to design the capacity allocated to minimize cost. To improve this, we decided to apply a new idea. We tuned the capacity allocation by adding multiples of the predicted traffic. This result shows quite saved traffic costs. It has an optimal point at k =0.9, which is quite effective. The average cost at this time is about 7.68 Gbyte when converted to penalty price with the asymmetric pricing structure. But this result is not efficient enough for the enterprise because the value k changes dramatically depend on peak traffic. The optimal point of k in this approach, k = [0.8, 5.6]. The value of k varies greatly, this model is not robust.

The result at this time is as follows.



- Result Capacity



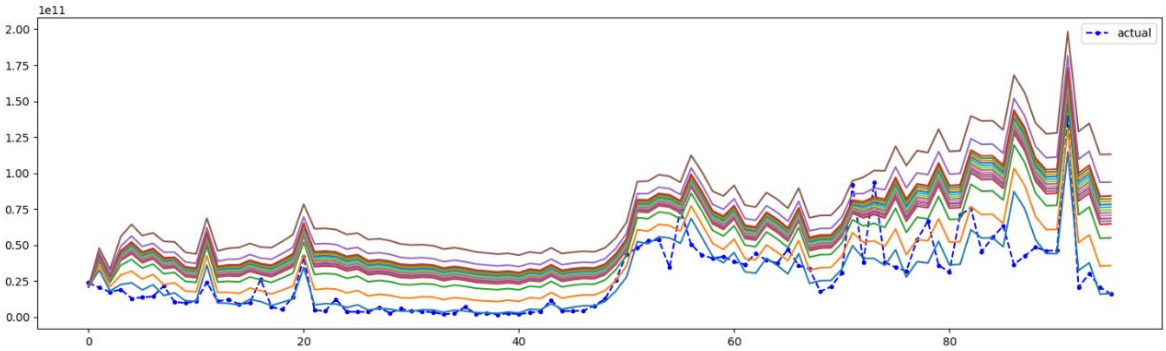
- Cost Change Flow

Fig. 7. Simple multiplication.

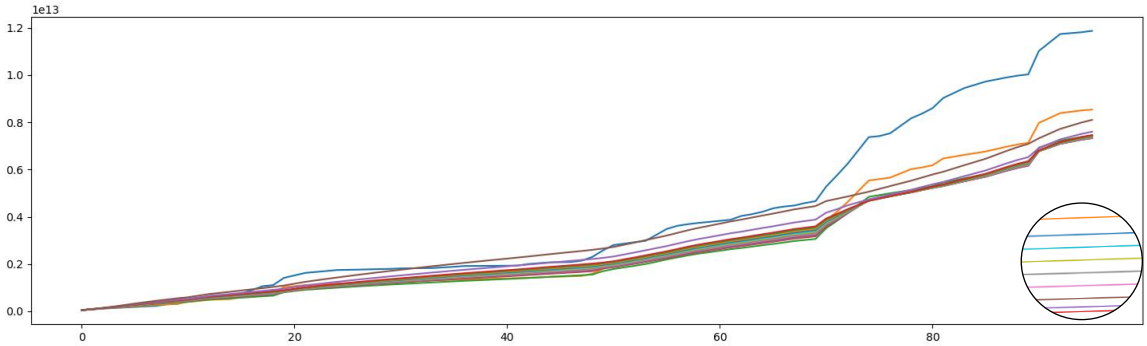
Another approach is to use standard deviation. The asymmetric cost structure allocates costs by the difference between actual traffic and predicted traffic. In other words, the cost varies greatly depending on the difference. The model would be improved by applying the standard deviation to traffic prediction in this case. In this approach, the standard deviation between actual and predicted traffic is multiplied by k times. This method is well matched during the upswing period, which

reduced network costs through STD usage. However, this approach has weak points that standard deviations remain due to abnormality points.

As a result of Fig. 8, we can see that it is quite different from actual traffic. The optimal point is at k =2.5 while k = {1.6 ~ 2.8} makes it generally optimal. Optimized cost is 7.63 Gbyte. The result at this time is as follows.



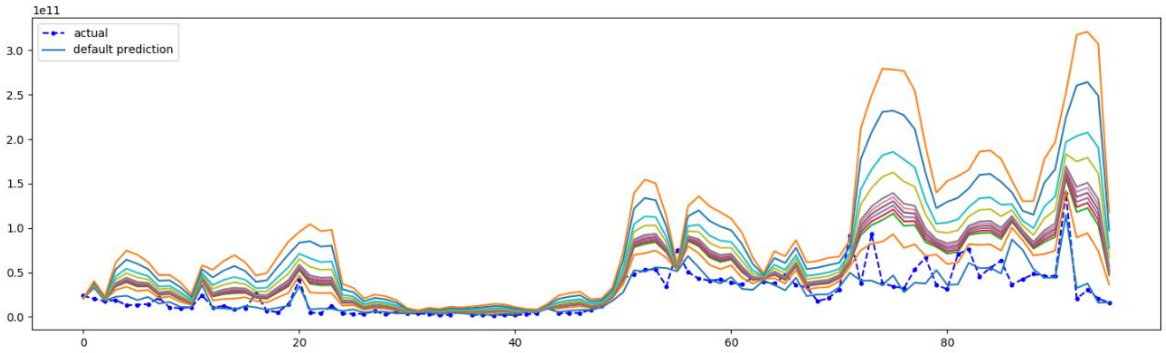
Result Capacity



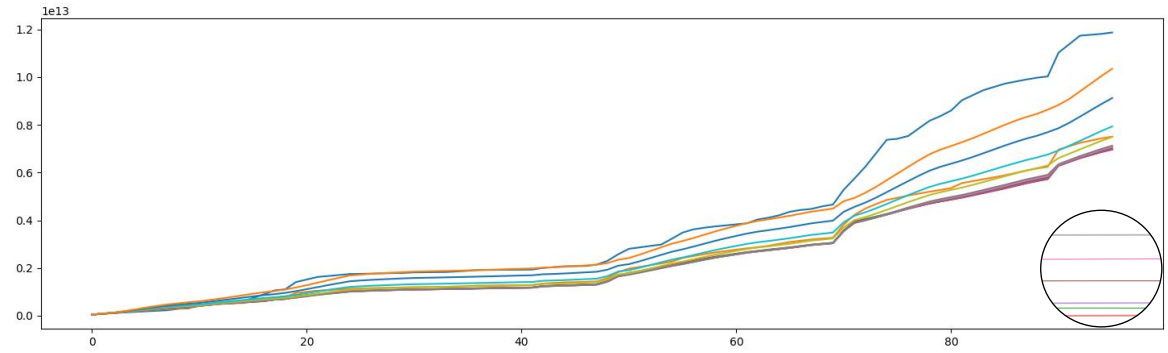
- Cost Change Flow

Fig. 8. Additional standard error.

We devise the limitation window at standard error method. In order to eliminate this peak traffic from standard deviation remaining, the standard deviation calculates only adjacent window. The effect of peak traffic was effectively lowered than the previous method, with the best window size of 3. The best performance at this time depends on the sample, but it is generally optimized at k = {1.5 ~ 3.8}. In this approach, best point at k=1.6. Optimized cost is 7.26 Gbyte. The result at this approach is shown in Fig.9.



- Result Capacity

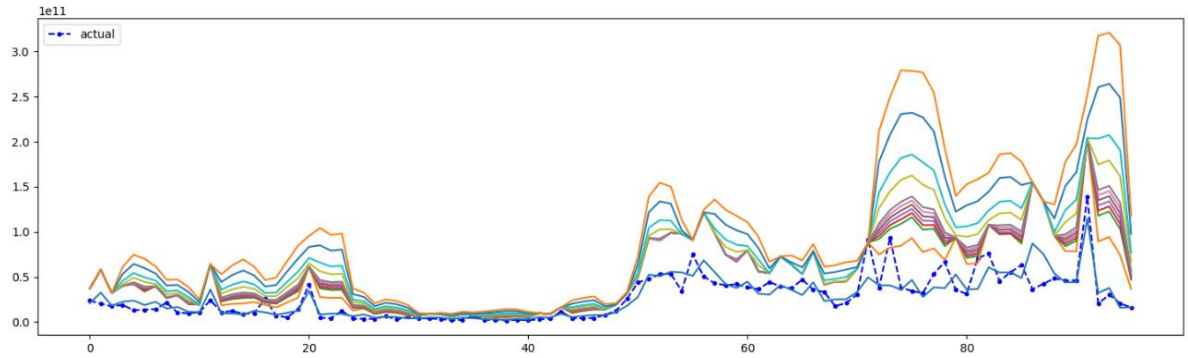


- Cost Change Flow

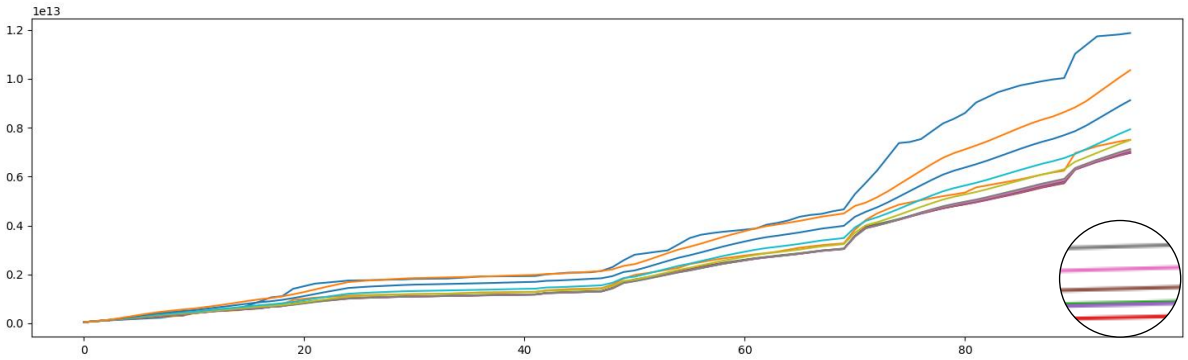
Fig. 9. Standard deviation with limited adjacent samples.

Based on our previous approach, we developed a heuristic approach. To minimize the cost, we apply the lower prediction bound to effectively raise the overall prediction. This method is the maximum between std_{W} and multiplication at $p^{0.25}$. As such, we have proposed a general lower bound, which is calculated as about $10^{0.25}$ the deep-learning prediction. This lower bound is also depending on the penalty cost, is effective for changing the penalty ratio. In addition, when the heuristic combination is applied, the value of k is changed to previous value. This case k value is optimal at k=1.6 and in the other time, k = {1.5~3.2}. The optimal value through this approach was 7.38 Gbyte.

The result at this time is as follows.



- Result Capacity



- Cost Change Flow

Fig. 10. Heuristic combination of multiplication and standard error ($w = 3, k$).

So far as we've been working this improvement, we find out that capacity allocation could reduce the network cost effectively. We compared the performance of the above five methods used in this study.

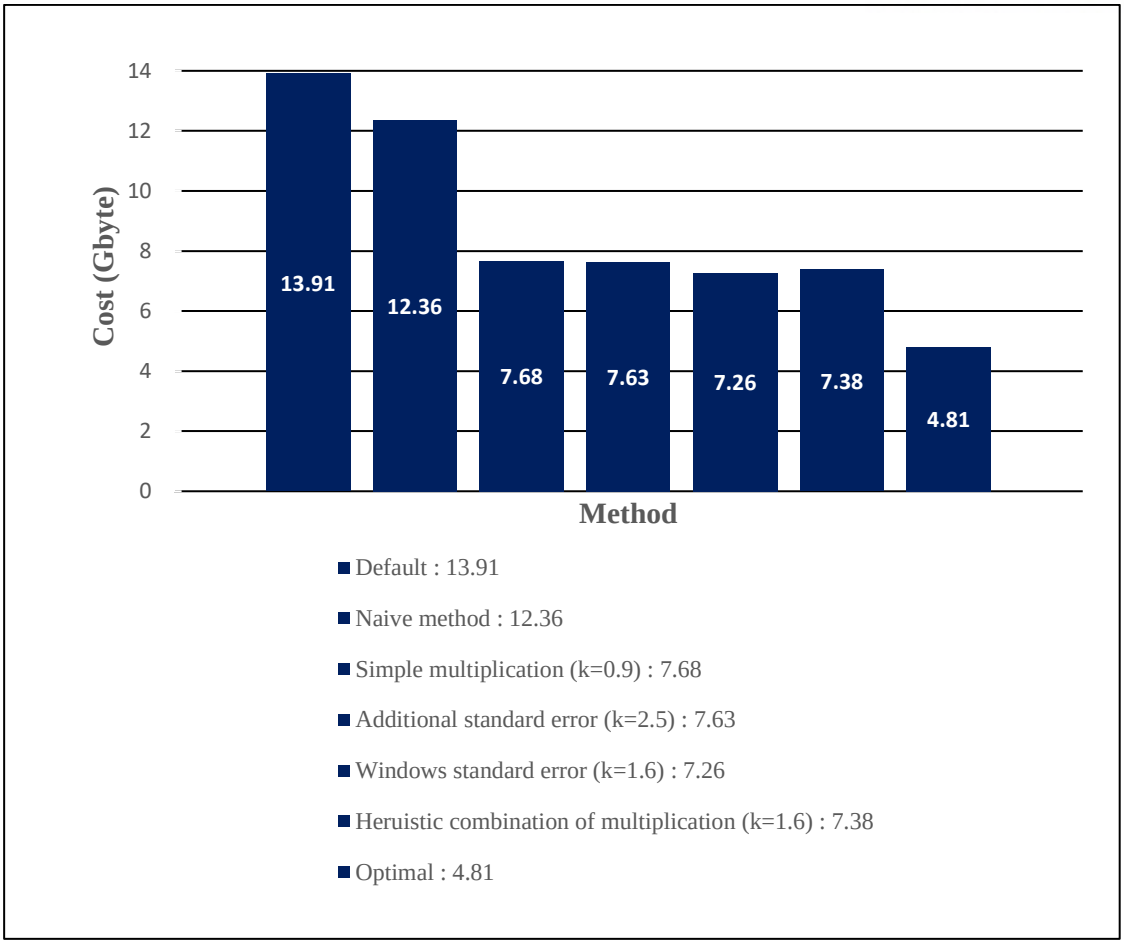


Fig. 11. Cost comparison of the proposed schemes (Gbytes).

Default value is in case of not adjusted capacity allocation. At that point, the traditional cost is 13.91 Gbyte. When the best performance adjusts the internet cost at 7.26 Gbyte. If the optimal cost (the minimum cost as the actual traffic) is applied, they are 4.81 Gbyte. Applying this study on a real-world traffic basis could save about 47.8%. If there has changed sequence, result is a nearly similar performance.

VI. CONCLUSION

We investigate the capacity cost minimization through traffic-predication-based dynamic capacity allocation. We have demonstrated that a firewall log dataset can be processed to predict the traffic by employing a deep-learning based approach (with an accuracy of 69.2%), by using the real-world dataset. Also, it is verified that appropriate capacity allocation scheme that takes into account the asymmetric pricing structure can improve the performance, by balancing the cost of prediction errors and the over-capacity overhead.

Our results are still preliminary in the following sense. We believe that there is still room for improvement in both prediction and cost minimization. Also, we only consider the scenario that the ISP accommodates the exceeding traffic beyond the requested capacity. In case that the ISP fails to support the excessive traffic, it will cause congestion and the cost associated with queueing or packet drops has to be accounted, which remains as an interesting future work.

REFERENCE

[1] Changhee Joo, Ness B Shroff, “A novel coupled queueing model to control traffic via QoS-aware collision pricing in cognitive radio networks”, IEEE INFOCOM 2017-IEEE Conference on Computer Communications, pp. 1-9, May 2017

[2] Sravan Patchala, Seung Hyun Lee, Changhee Joo, D Manjunath, “On the Economics of Network Interconnections and Net Neutrality”, 2019 11th International Conference on Communication Systems & Networks (COMSNETS), pp.192-199, January 2019

[3] Luca Chiaraviglio, Marco Mellia, and Fabio Neri, “Minimizing ISP Network Energy Cost: Formulation and Solutions”, IEEE/ACM Transactions On Networking, Vol. 20, No. 2, April 2012

[4] Warren B. Powell, Patrick Jaillet and Arnedeo Odoni, “Stochastic and Dynamic Networks and Routing”, Handbooks in Operations Research and Management Science, Vol 8, pp 141-295, 1995

[5] Dulakshi S. K. Karunasinghea, Shie-Yui Liong, “Chaotic time series prediction with a global model: Artificial neural network”, Journal of Hydrology, Vol 323, Issues 1–4, pp.92-105, May 2006

[6] Weitao Wang, Yuebin Bai, Chao Yu, Yuhao Gu, Peng Feng, Xiaojing Wang and Rui Wang “A network traffic flow prediction with deep learning approach for large-scale metropolitan area network”, NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, April 2018.

[7] D. Morato, J. Aracil, L.A. Diez, M. Izal and E. Magana, “On linear prediction of Internet traffic for packet and burst switching networks” ICCCN, 2001.

[8] Aimin Sang, San-qi Li, “A predictability analysis of network traffic” Computer Networks, ELSEVIER, Volume 39, Issues 4, pp.329–345, July 2002.

[9] S. Chong, S-Q. Li, J. Ghosh, “Dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM” Proceedings of INFOCOM '94 Conference on Computer Communication. 06 August 2002.

[10] Jim Gao, “Machine Learning Applications for Data Center Optimization”, Google, 2014.

[11] Retrieved from “https://www.cisco.com/c/m/en_us/techdoc/dc/reference/cli/nxos/commands/l2/bandwidth-interface.html”

[12] Retrieved from “https://www.cisco.com/c/en/us/td/docs/solutions/CVD/Campus/sda-sdg-2019oct.html”

[13] Kevin P. Murphy “Machine Learning: A Probabilistic Perspective”, MIT, pp.565-566, 2012.

[14] Manish Joshi, Theyazn Hassn Hadi, “A Review of Network Traffic Analysis and Prediction Techniques” [online] Available: https://arxiv.org/abs/1507.05722, 2015.

[15] Samira Chabaa, Abdelouhab Zeroual, and Jilali Antari, “Identification and Prediction of Internet Traffic Using Artificial Neural Networks”, J. Intelligent Learning Systems & Applications, pp.147-155. 2010.

[16] Paulo Cortez, Miguel Rio, Miguel Rocha, and Pedro Sousa, “Internet Traffic Forecasting using Neural Networks”, International Joint Conference on Neural Networks, pp.2635-2642. Vancouver, Canada, 2006.

[17] S. Trenn, “Multilayer perceptrons: approximation order and necessary number of hidden units,” IEEE Transactions on Neural Networks, vol. 19, no. 5, pp. 836–844, 2008.

[18] Dinh, Laurent, Krueger, David, and Bengio, Yoshua. “NICE: Non-linear independent components estimation” arXiv preprint arXiv:1410.8516, 2014.

[19] M. Gardner, S. Dorling, "Artificial neural networks (the multilayer perceptron): A review of applications in the atmospheric sciences", Atmos. Environ., vol. 32, no. 14–15, pp. 2627-2636, Aug. 1998.

[20] X. Glorot, A. Bordes, Y. Bengio, "Deep Sparse Rectifier Neural Networks", Proc. Conf. Artificial Intelligence and Statistics, 2011.

[21] Pierre Baldi, Peter J Sadowski, C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K.Q. Weinberger, "Understanding dropout" in Advances in Neural Information Processing Systems 26, Curran Associates, Inc, pp. 2814-2822, 2013

ACKNOWLEDGEMENT

I feel very appreciate to my master paper committee, and especially appreciate to professor Joo. I could not achieve this paper without his teaching. And thanks to my family and friends for supporting. Also I greatly thank my company colleagues for worked hard till my absence.